DISCUSSION DOCUMENT

Information on Submitted Manuscript

- Title: Deep Learning Approach for Physical Layer Security in Gaussian Multiple Access Wire-tap Channel
- Author: Emmanuel Obeng Frimpong, Taehoon Kim and Inkyu Bang

SUMMARY

In this document, we discuss the details of extra simulations and results for our manuscript. Specifically, we take at look at our system performance when we do the following:

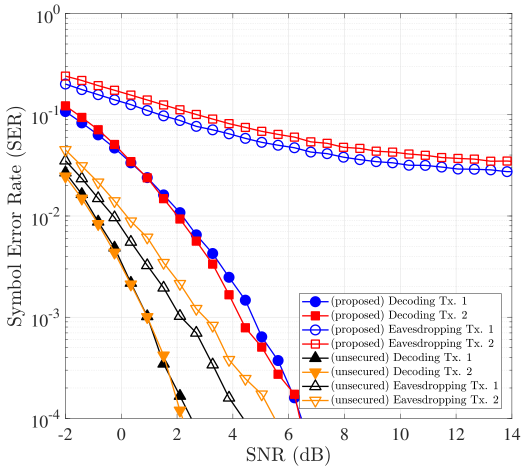
- Increase layers of nodes (Alice, Bob and Eavesdropper) in our system model.
- Increase number of transmit nodes
- Integrate channel coding in our training procedure

1 Increasing the Number of Layers

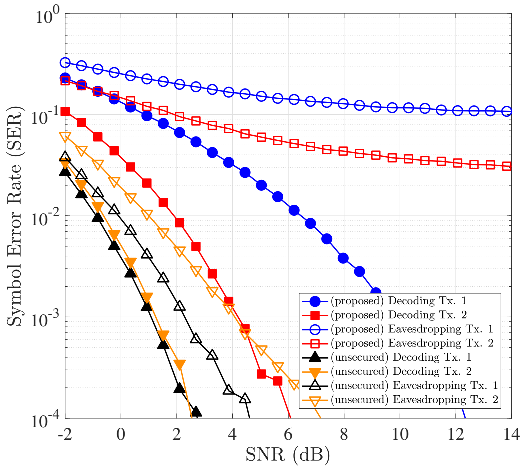
In the section we review the effect of adding extra layers to make a larger model for Alice and Bob. We have increased the layers to 5 and 7 to see if the performance will improve. We considered both asymmetric increment where entities in the network have different number of layers and symmetric increment of layers where all parties (transmitters, legitimate receivers and eavesdropper) have the same number of layers. Firstly, for asymmetric, we increase the dense layers from 3 to 5, and 7 for both transmitters only. And then, we also increase for both transmitters and legitimate receiver from 3 to 5, and 7. For symmetric increment, we increase layers of transmitters, legitimate receiver and eavesdropper to 5 and also to 7. We observed that when we increase layers of all entities (symmetric increment) to 5 and 7, system performance is similar to having 3 layers. Also asymmetric increment slightly improves system performance, since we inherently create an unfair advantage. However, for both cases, further increase in the number of layers might cause our model to overfit and perform poorly.

1.1 Performance Evaluation

In the section we discuss the simulation results for each layer increment we considered. We used the same training procedure and parameters discussed in our manuscript.
In Fig. A, we perform asymmetric increment by increasing the layers of only the encoders while maintaining 3 layers for legitimate decoder and eavesdropper. We observe that as we increase the layers from 3 to 5 in Fig. Aa and to 7 in Fig. Ab, the secrecy performance of the system becomes better as the symbol error rate (SER) of the eavesdropper remains high at even high SNR. This is because increasing the only the layers of encoders creates an unfair advantage such that encoders are able to efficiently learn secure encoding rules to deceive the eavesdropper. However, as we increase the encoder layers from 3 to 5 and to 7, the performance of the legitimate decoder becomes poor.



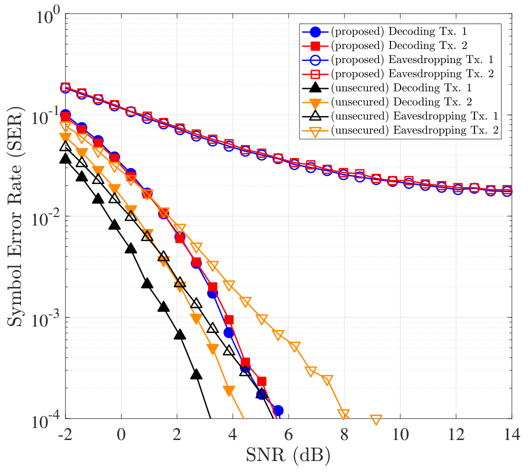
(a) Only encoders with layers increased to 5



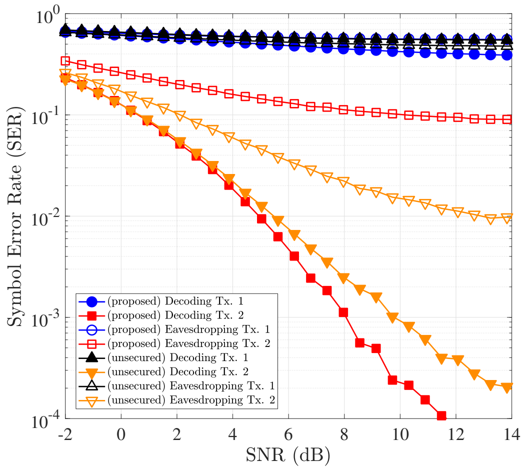
(b) Only encoders with layers increased to 7

Figure A: Increasing layers for only encoders

Fig. B shows another asymmetric increment for the encoders and the legitimate decoder only. In Fig. Ba, we increase layers of the encoders and legitimate receiver to 5 and 7 in Fig. Bb, while eavesdropper has 3 layers. We observe in Fig. Ba that the secured performance against eavesdropper remains almost the same as when all entities had 3 layers. However, the legitimate receiver is able to decode secured symbols efficiently compared to previous 3 layers. In Fig. Bb, increasing layers of encoders and legitimate decoders causes our model to overfit and performs poorly.



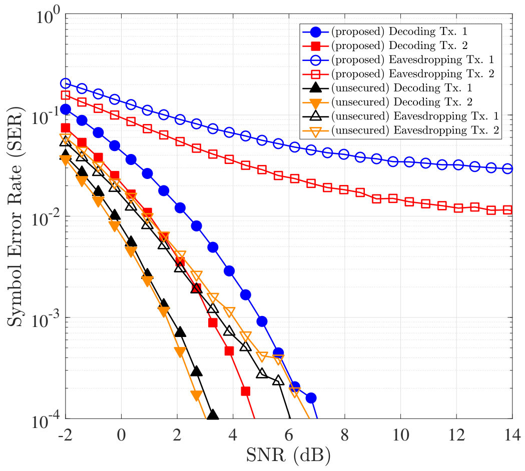
(a) Only encoders and legitimate receiver with layers increased to 5



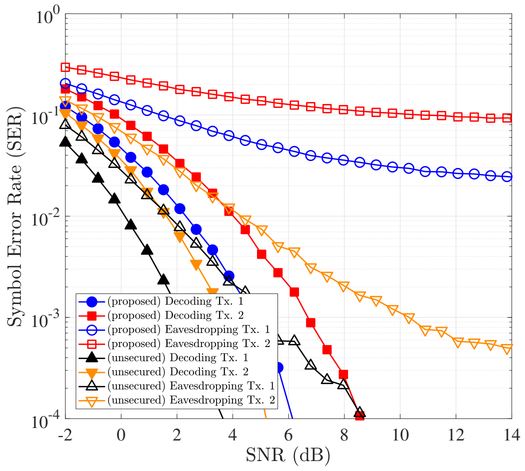
(b) Only encoders and legitimate receiver with layers increased to 7

Figure B: Increasing layers for Both encoders and legitimate receiver only

Fig. C shows the SER performance for symmetric increment where all entities (encoders, legitimate decoder and eavesdropper) have same number of layers. We observe that in Fig. Ca, where all entities have 5 layers, system performance is almost similar to when entities had 3 layers. However, SER performance degrade at legitimate receiver for decoding unsecured symbols. In Fig. Cb, all entities have 7 layers. We observe that it is difficult for eavesdropper to decode secured symbols. However, the performance of the legitimate receiver in decoding both secured and unsecured transmissions degrade compared to when all entities have 3 layers.



(a) Encoders, legitimate receiver and eavesdropper with layers increased to 5



(b) Encoders, legitimate receiver and eavesdropper with layers increased to 7

Figure C: Increasing layers for encoders legitimate receiver and eavesdropper

2 Three-User Setup

In this section, we simulate for a three-user multiple access wiretap system with larger block size $n = 32$. We discuss the model, parameters and the simulation results. We considered $K = 3$ transmitters, a legitimate receiver and an eavesdropper Fig. D.

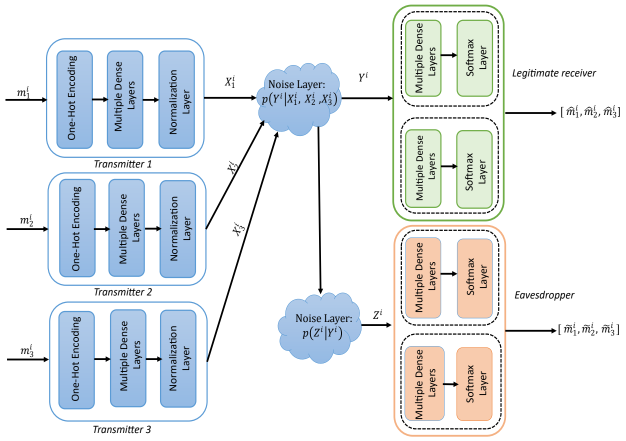


Figure D: System Model for 3 Users

2.1 Simulation and Result Discussion

First we simultaneously train the transmitters and the first logical module of the legitimate receiver using (2), the linear combination of the standard cross-entropy (1).

$$L_R\left(\mathbf{s}_k, \hat{\mathbf{s}}_k\right) \triangleq -\sum_{m=1}^{|\mathcal{M}|} s_{k,m} \log\left(\hat{s}_{k,m}\right) \tag{1}$$

We express the linear combination as

$$L = \sum_{k=1}^{K} \alpha_k L_{R_k}\left(\mathbf{s}_k, \hat{\mathbf{s}}_k\right) \tag{2}$$

We performed hyperparameter tuning for best performance and th selected values we $\alpha_1 = \alpha_2 = \alpha_3 = 6$. We now train the second logical module of the legitimate receiver with (2) and tuned values $\alpha_1 = \alpha_2 = \alpha_3 = 10$. Now, transmitter and legitimate receiver have learned the encoding and decoding rules. Then, we train the first logical module of the eavesdropper to be able to decode the encoding rules that has been previously learned by the transmitters. During this training, we used

(4) the linear combination of the standard cross-entropy (3).

$$L_E\left(\mathbf{s}_k, \tilde{\mathbf{s}}_k\right) \triangleq -\sum_{m=1}^{|\mathcal{M}|} s_{k,m} \log\left(\tilde{s}_{k,m}\right), \tag{3}$$

We express the linear combination as

$$L = \sum_{k=1}^{K} \alpha_k L_{E_k}\left(\mathbf{s}_k, \tilde{\mathbf{s}}_k\right) \tag{4}$$

After several tuning, $\alpha_1 = \alpha_2 = \alpha_3 = 4$ were the optimal values used during the training. Then we repeat the training for the second logical model of the eavesdropper using (4) with $\alpha_1 = \alpha_2 = \alpha_3 = 5$. For the secure training, first we simultaneously train transmitters and the first logical module of the legitimate receiver with the secured loss function (5).

$$L = \alpha_1 \left[\sum_{k=1}^{K} \beta_k L_R(\mathbf{s}_k, \hat{\mathbf{s}}_k)\right] - \alpha_2 \left[\sum_{k=1}^{K} \gamma_k L_E(\mathbf{s}_k, \tilde{\mathbf{s}}_k)\right] \tag{5}$$

Hyperparameter values that gave the best performance after tuning is $\alpha_1 = 1, \alpha_2 = 10$, $\beta_1 = \beta_2 = \beta_3 = 6$ and $\gamma_1 = \gamma_2 = \gamma_3 = 10$. Then using (5) and parameters $\alpha_1 = 1, \alpha_2 = 15$, $\beta_1 = \beta_2 = \beta_3 = 5$ and $\gamma_1 = \gamma_2 = \gamma_3 = 10$, we trained the transmitters and the second logical module of the legitimate receiver. The purpose of the secured training is for transmitters to learn symbol representation that increases loss at the eavesdropper side. After, we test and show the results in Fig. E.

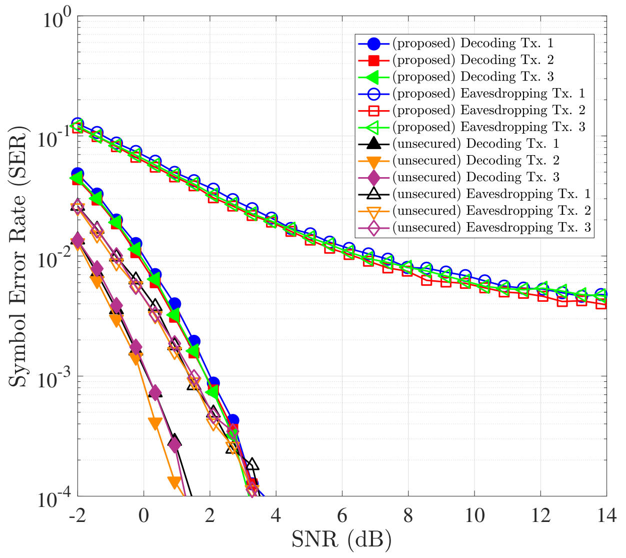


Figure E: SNR Versus Symbol Error Rate for 3 Users

3 Channel Coding

In this section, we discussion the implementation of channel coding in our model and discuss the simulation results.
It is known that transmission rate is dependent on the mutual information between the input X and output Y of a channel. Therefore the channel capacity is

$$C = \max_{p} I(X;Y) \tag{6}$$

where the *max* is taken over the possible input probabilities p. This implies that mutual information can be used as a measure to learn the channel coding function of the wireless channel as used in [R1]. The mutual information is given by

$$\begin{aligned} I(X;Y) &= \int_{X\chi Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)}\, dx\, dy \\ &= D_{KL}(p(x,y)||p(x)p(y)) \\ &= \mathbb{E}_{p(x,y)}\left[log \frac{p(x,y)}{p(x)p(y)} \right] \end{aligned} \tag{7}$$

Authors in [R2] introduced Mutual Information Neural Estimation which implements the Donsker-Varadhan representation of Kullback-Leibler (KL) divergence and based on these, proposed a neural network (NN) that estimates the lower bound of mutual information. The Donsker-Varadhan representation of KL divergence can be stated as

$$D_{KL}(P||Q) = \sup_{g} \mathbb{E}_{p}\left[g(X,Y)\right] - log\left(\mathbb{E}_{Q}[e^{g(X,Y)}]\right) \tag{8}$$

The lower bound mutual estimator yields

$$I(X;Y) \geq \sup_{\theta \in \Theta} \mathbb{E}_{p(x,y)}\left[T_{\theta}(X,Y)\right] - log \mathbb{E}_{p(x)p(y)}[e^{T_{\theta}(X,Y)}] \tag{9}$$

Proof in [R2]
We integrate the NN mutual information estimator on our framework. This technique and its variants have been adopted in several works as [R1, R3–R5]. We maximize the mutual information between the samples of the channel input and output and optimize the weights of he encoders. We maximize equation (9) with Adam optimizer. Since we do not have the joint distribution $p(x,y)$ and marginal distributions, we use samples of these distributions and approximate the expectations by the sample average.
For k samples

$$I_{\theta}(X^n;Y^n) = \frac{1}{k}\sum_{n=1}^{k}\left[T_{\theta}(x_i^n, y_i^n)\right] - log \frac{1}{k}\sum_{n=1}^{k}[e^{T_{\theta}(x_i^n, \tilde{y}_i^n)}] \tag{10}$$

We send the generated m messages through each encoder which produces X^n. Then we pass X^n through the channel to produce corresponding Y^n. This forms the first term in equation (10). For the second term in equation (10), we follow procedure in [R1, R2] by dropping either x_i^n or y_i^n from the joint sample (x_i^n, y_i^n) and dropping the other in next k sample. In equation (10), T_θ represents a neural network with 2 fully connected hidden layers with 256 hidden nodes and a linear output node.

The training of encoders is implemented in 2 phases. We train both encoders separately and for each encoder we implement an NN estimator. In phase 1, we initiate the encoders and train the mutual information estimation networks for both encoders. Mutual information estimator networks for encoder 1 and encoder 2 are trained with same parameters: 500 iterations, batch size 64 and learning rate 0.006. Then in the second phase, we minimize 10 over each encoders weights with their respective estimators. For this also, we sued same parameters for both encoders: 400 iterations, batch size 64 and learning rate 0.005. Then we continue with the unsecured training and secured training as described in Section 3.3 (Training Phase) of our manuscript. We then plot and see the impact of channel coding on our system performance. Then we compare with results that did not consider channel coding in Fig. F."

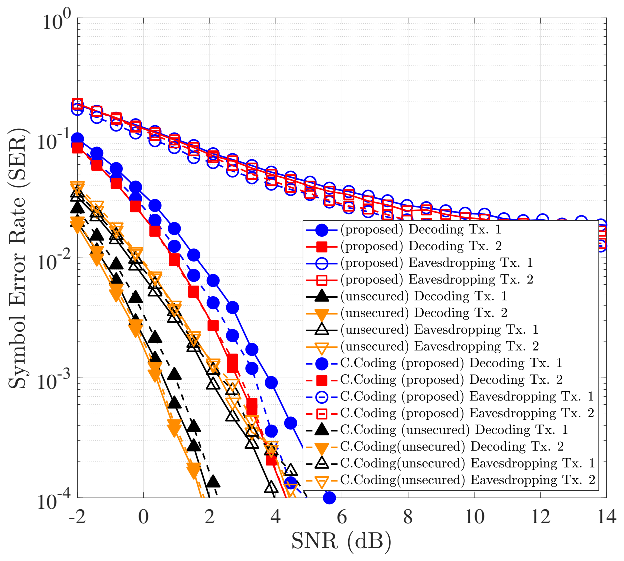


Figure F: Error rate for Training with Channel Coding and without Channel Coding

References

[R1] R. Fritschek, R. F. Schaefer, and G. Wunder, "Deep learning for channel coding via neural mutual information estimation," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2019, pp. 1–5.

[R2] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "Mine: mutual information neural estimation," *arXiv preprint arXiv:1801.04062*, 2018.

[R3] S. Molavipour, G. Bassi, and M. Skoglund, "Conditional mutual information neural estimator," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 5025–5029.

[R4] Q. Wu, J. Tang, S. Dang, and G. Chen, "Data privacy and utility trade-off based on mutual information neural estimator," *Expert Systems with Applications*, vol. 207, p. 118012, 2022.

[R5] X. Lin, I. Sur, S. A. Nastase, A. Divakaran, U. Hasson, and M. R. Amer, "Data-efficient mutual information neural estimator," *arXiv preprint arXiv:1905.03319*, 2019.